EGE ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Database Management

2022-2023 Term Project Report

HAZIRLAYANLAR

05200000788 Ahmet Burak Bilgiç

05190000106 Mustafa Duhan Boblanlı

05190000019 Barış Bektaşoğlu

05200000099 Arda Bozkurt

1. Write a brief explanation using your own words (in English) about the given design.

This Entity Relationship diagram belongs to a database used by a university. There are entities that reflect the real world such as COLLEGE, INSTRUCTOR, DEPT, COURSE, SECTION, STUDENT. There are also relationships that these entities have like DEPT --> EMPLOYS --> INSTRUCTOR, STUDENT --> TAKES --> SECTION and the minimum and maximum number of constituents in the given relationship.

Each entity has multiple attributes that differ based on its needs. For example an INSTRUCTOR has attributes like Id, Rank, IName, IOffice, IPhone and a unique attribute is used to identify each entity. In this case the unique identifier used to identify a specific INSTRUCTOR is Id.

2. Write an analysis report:

a. What is the aim of your design?
   The primary aim of this design is to satisfy the new requirements given. In addition to this, this new EER diagram helps make the former ER diagram more readable and tidy. With the additions, the design became more understandable and rich in meaning. Another priority taken into consideration is trying our best to not interfere with the previous requirements and the structures in the former design while introducing and integrating the new requirements.

b. What are the main entities?
   COLLEGE
   DEPT
   COURSE
   SECTION
   STUDENT
   INSTRUCTOR

c. What are the characteristics of each entity?
   i. College is an entity that administers departments. It serves an organisational purpose when it comes to keeping track of and managing departments.
   ii. Department serves as a central entity that helps facilitate multiple critical functionality. It is related to many other entities such as college, instructor, student and course. It assumes a critical mission when it comes to the interaction between mentioned entities.
   iii. Course is an entity that stores information about each course taught.
   iv. Section is an entity that is used in reference to course. Semantically, it is often used to allot a timetable into parts. This entity is also used to establish the connection between an instructor and the courses taught by that instructor. Between student entity and section entity, there exists an interesting relationship that reflects the domain.
   v. Student entity is used to facilitate the main aim of the organisation. Each student participates in sections and gets grades. This entity is abstracted from other complexities of courses and other entities. It only interacts with sections and departments.
   vi. Instructor is the entity that represents an integral part of this organisation. It is as crucial as the student to the main facility this school system aims to offer. Instructors participate in activities such as teaching and managing.

d. What relationships exist among the entities?
   i. *Dean*: between College and Professor
   ii. Admins: between College and Dept
   iii. Chair: between Dept and Instructor
   iv. Follows: between Dept and Curriculum
   v. Has: between Dept and Person
   vi. Takes: between Student and Section
   vii. Secs: between Course and Section
   viii. Assists: between Research Assistant and Section
   ix. Teaches:between Instructor and Section
   x. Has: between Course and Curriculum

e. What are the constraints related to entities, their characteristics and the relationships among them?

   i. • Only professors can hold the title of chair or dean.
   ii. • A section can only be given by one lectures
   iii. • Different types of curriculums can include the same course.

iv. • A lecture can teach in a department to which he or she is not attached.
v. • Each research assistant must have a bachelor's degree.
vi. • Any classroom in a department can only be used for one session at a given
vii. • A student cannot take courses from another department other than her own department.
viii. • Lecturer can only be in one place at the same time during the same period of

3. Create an EER diagram. Try to use enhanced/extended features of ER modelling. Do not use any tool. You can use any drawing application with the right legend for ER modelling. The output of this step is just an EER diagram.

The EER Diagram is in its own file since it is too large to be inserted here.

4. The most important point of your design is how to extend the original design and generate added value. Therefore, you should accurately examine the extensions to the original design. You should determine the interaction points of the newly added requirements. You can define new entities where interaction and integration are required. At this point your creativity has an artistic significance.

The EER Diagram is in its own file since it is too large to be inserted here.

5. Write down the data requirements for the EER diagram.

COLLEGE has a unique CName that is used as an identifier, a COffice and a CPhone.
A College ADMINS N DEPTs or may not ADMIN any DEPT.
A College DEANs 1 and only 1 Professor.

DEPT has a unique DName, a unique DCountry, a unique DCode that is used as an identifier; also has a DOffice and a DPhone.
A DEPT must be ADMINistrated by 1 and only 1 COLLEGE.
A DEPT FOLLOWS N CIRRICULUMs or may not FOLLOW any CURRICULUM.
A DEPT HAS N PERSONs or at least 1 PERSON.
A DEPT CHAIRS 1 and only 1 INSTRUCTOR.

CURRICULUM has a unique Cu_id that used as an identifier.
A CURRICULUM must be FOLLOWed by 1 and only 1 DEPT.
A CURRICULUM HAS N COURSEs or may not HAS any COURSE.

COURSE has a unique CCode, a unique CoName that is used as an identifier; also has a Credits, a Level, a CDesc and Keyword/Keywords.
COURSE has selection,COURSE either must be a MANDATORY or must be an OPTIONAL.
COURSE SECS N SECTIONs or may not SEC any SECTION.

OPTIONAL has selection, either OPTIONAL must be a TECHNICAL or must be a NONTECHNICAL.

PERSON has a unique PId that is used as an identifier, a PName that has a FName and a LName, a Addr and A Phone.
PERSON must HAS 1 and only 1 DEPT.
PERSON has selection,PERSON either must be a STUDENT or must be a FACULTY MEMBER.

STUDENT has a Major and a DOB.
STUDENT TAKES N SECTIONs OR MAY NOT TAKE any SECTION.

SECTION has a unique SecId that is used as an identifier, a Sem, A Year, a SecNo, A DaysTime and a CRoom that has a Bldg and RoomNo.
SECTION must be SECd by 1 and only 1 COURSE.
SECTION TEACHES 1 and only 1 INSTRUCTOR.
SECTION ASSIST N RESEARCH ASSISTANT or may not ASSIST any RESEARCH ASSISTANT.

FACULTY MEMBER has a M.Sc, A Ph.D, EOffice and ResearchArea/ResearchAreas.
FACULTY MEMBER has selection, FACULTY MEMBER either must be a RESEARCH ASSISTANT or must be an INSTRUCTOR.

RESEARCH ASSISTANT must ASSIST N SECTIONs or may not ASSIST any SECTION.

INSTRUCTOR must CHAIR a DEPT or may not CHAIR any DEPT.
INSTRUCTOR has selection, INSTRUCTOR either must be a PROFESSOR or must be an ASSOCIATE PROFESSOR or must be an ASSISTANT PROFESSOR.

PROFESSOR must DEAN a COLLEGE or may not DEAN any COLLEGE.

6. Convert EER diagram into relational model using the methodology that will be introduced in your course.

1st iteration:
Step 1:
DEPT(DName, 'DCode', DCountry, DOffice, DPhone)
COLLEGE('CName', COffice, CPhone)
CURRICULUM('Cu_id')
COURSE('CCode', CoName, Credits, Level, CDesc)
SECTION('SecId', SecNo, Bldg, RoomNo, DaysTime, Year, Sem)
PERSON('PId', FName, LName, Addr, Phone)
Step2:
Step3:
Step4:
DEPT(DName, 'DCode', DCountry, DOffice, DPhone, COLLEGE.CName.ADMIN_ColName)
CURRICULUM('Cu_id', DEPT.DCode.FOLLOWS_DepCode)
COURSE('CCode', CoName, Credits, Level, CDesc, CURRICULUM.Cu_id.HAS_Cur_id)
SECTION('SecId', SecNo, Bldg, RoomNo, DaysTime, Year, Sem, COURSE.CCode.CCode)
PERSON('PId', FName, LName, Addr, Phone, DEPT.DCode.DCode)
Step5:
Step6:
COURSE_KEYWORD('DCode, Keyword')
Step7:
Step8:
// using 8.a
COURSE('CCode', CoName, Credits, Level, CDesc, CURRICULUM.Cu_id.HAS_Cur_id) // no changes
MANDATORY_COURSE('COURSE.CCode')
OPTIONAL_COURSE('COURSE.CCode')

// using 8.b
PERSON table is deleted
FACULTY_MEMBER('PId', FName, LName, Addr, Phone, DEPT.DCode.DCode, EOffice, Ph.D, M.Sc)
STUDENT('PId', FName, LName, Addr, Phone, DEPT.DCode.DCode, Major, DOB)
Step 9:
2nd Iteration:

Step1-4:
Step5:
STUDENT_TAKES_SECTION('STUDENT.PID,SECTION.SecID', Grade)
Step6:
FAC_MEM_RESEARCH_AREAS('FACULTY_MEMBER.PId, ResearchArea')
Step7:
Step8:
// using 8.c
OPTIONAL_COURSE('COURSE.CCode', isTechnical)
// using 8.a
FACULTY_MEMBER('PId', FName, LName, Addr, Phone, DEPT.DCode.DCode, EOffice, Ph.D, M.Sc) // stays the same
RESEARCH_ASSISTANT('FACULTY_MEMBER.PId')
Instructor('FACULTY_MEMBER.PId')
Step 9:

3rd Iteration:

Step1-2:
Step3:
DEPT(DName, 'DCode', DCountry, DOffice, DPhone, COLLEGE.CName.ADMIN_ColName, Instructor.PId.ChairId, ChairsStartDate)
Step4:
SECTION('SecId', SecNo, Bldg, RoomNo, DaysTime, Year, Sem, COURSE.CCode.CCode, Instructor.PId)
Step5:

RESEARCH_ASSISTANT_ASSISTS_SECTION('FACULTY_MEMBER.PId, SecId')
INSTRUCTOR_MATCHES_COURSE('FACULTY_MEMBER.PId, CCode', CoMatchRate)
Step6-7:
Step8:
// using 8.c
INSTRUCTOR('FACULTY_MEMBER.PId', ProfType) //Proftype is a string
Step 9:

4th Iteration:
Step1-2:
Step3:
COLLEGE('CName', COffice, CPhone, INSTRUCTOR.PId.DeanId) // ProfType must be 'PROFESSOR'
Step4-9:

7. Write down the appropriate SQL scripts (DDL statements) for creating the database and its relational model. You can select any of the DBMS you wish.

```sql
DROP DATABASE IF EXISTS university_db_schema;
CREATE SCHEMA university_db_schema;

USE university_db_schema;

CREATE TABLE university_db_schema.dept (
   d_code INT PRIMARY KEY,
   d_name VARCHAR(255) NOT NULL,
   d_country VARCHAR(255) NOT NULL,
   d_office VARCHAR(255) NOT NULL,
   d_phone VARCHAR(255) NOT NULL,
   admin_col_name VARCHAR(255) NOT NULL,
   chair_id INT NOT NULL,
   chair_start_date DATE NOT NULL
);

CREATE TABLE university_db_schema.faculty_member (
   p_id INT PRIMARY KEY,
   f_name VARCHAR(255) NOT NULL,
   l_name VARCHAR(255) NOT NULL,
   address VARCHAR(255) NOT NULL,
   phone VARCHAR(15) NOT NULL,
   d_code INT NOT NULL,
   e_office VARCHAR(50) NOT NULL,
   msc VARCHAR(255) NOT NULL,
   phd VARCHAR(255) NOT NULL
);

CREATE TABLE university_db_schema.prof_types (
   prof_type VARCHAR(45) PRIMARY KEY
);
INSERT INTO university_db_schema.prof_types(prof_type)
VALUES ('PROFESSOR'), ('ASSOCIATE PROFESSOR'), ('ASSISTANT PROFESSOR');

CREATE TABLE university_db_schema.instructor (
   p_id INT PRIMARY KEY,
   prof_type VARCHAR(45) NOT NULL
);

CREATE TABLE university_db_schema.college (
   c_name VARCHAR(255) PRIMARY KEY,
```

```
   c_office VARCHAR(255) NOT NULL,
   c_phone VARCHAR(255) NOT NULL,
   dean_id INT NOT NULL
);

CREATE TABLE university_db_schema.curriculum (
  cu_id INT PRIMARY KEY,
  followed_by_deptCode INT NOT NULL
);

CREATE TABLE university_db_schema.course (
   c_code INT PRIMARY KEY,
   co_name VARCHAR(255) NOT NULL,
   credits INT NOT NULL,
   course_level INT NOT NULL,
   c_desc VARCHAR(255) NOT NULL,
   in_curriculum INT NOT NULL
);

CREATE TABLE university_db_schema.course_keyword (
   c_code INT,
   keyword VARCHAR(100),
   PRIMARY KEY (c_code, keyword)
);

CREATE TABLE university_db_schema.mandatory_course (
   c_code INT PRIMARY KEY
);

CREATE TABLE university_db_schema.optional_course (
   c_code INT PRIMARY KEY,
   is_technical BOOLEAN NOT NULL
);

CREATE TABLE university_db_schema.student (
   p_id INT PRIMARY KEY,
   f_name VARCHAR(255) NOT NULL,
   l_name VARCHAR(255) NOT NULL,
   address VARCHAR(255) NOT NULL,
   phone VARCHAR(15) NOT NULL,
   d_code INT NOT NULL,
   major VARCHAR(100) NOT NULL,
   dob DATE NOT NULL
);

CREATE TABLE university_db_schema.student_takes_section (
   p_id INT NOT NULL,
    sec_id INT NOT NULL,
    grade INT,
    PRIMARY KEY(p_id, sec_id)
);

CREATE TABLE university_db_schema.fac_mem_research_areas (
   p_id INT NOT NULL,
    research_area VARCHAR(100) NOT NULL,
    PRIMARY KEY (p_id, research_area)
```

```
);

CREATE TABLE university_db_schema.research_assistant (
   p_id INT PRIMARY KEY
);

CREATE TABLE university_db_schema.section (
   sec_id INT PRIMARY KEY,
   sec_no INT NOT NULL,
   bldg VARCHAR(100) NOT NULL,
   room_no INT NOT NULL,
   DaysTime VARCHAR(100) NOT NULL,
   sec_year YEAR NOT NULL,
   sem VARCHAR(15) NOT NULL,
   c_code INT NOT NULL,
   instr_p_id INT NOT NULL
);

CREATE TABLE university_db_schema.research_assistant_assists_section (
   p_id INT NOT NULL,
   sec_id INT NOT NULL,
   PRIMARY KEY(p_id, sec_id)
);

ALTER TABLE dept
ADD FOREIGN KEY (admin_col_name) REFERENCES college(c_name),
ADD FOREIGN KEY (chair_id)REFERENCES instructor(p_id);

ALTER TABLE faculty_member
ADD FOREIGN KEY (d_code) REFERENCES dept(d_code);

ALTER TABLE instructor
ADD FOREIGN KEY (prof_type) REFERENCES prof_types(prof_type),
ADD FOREIGN KEY (p_id) REFERENCES faculty_member(p_id);

ALTER TABLE college
ADD FOREIGN KEY (dean_id) REFERENCES instructor(p_id);

ALTER TABLE course
ADD FOREIGN KEY (in_curriculum) REFERENCES curriculum(cu_id);

ALTER TABLE mandatory_course
ADD FOREIGN KEY (c_code) REFERENCES course(c_code);

ALTER TABLE optional_course
ADD FOREIGN KEY (c_code) REFERENCES course(c_code);

ALTER TABLE student
ADD FOREIGN KEY (d_code) REFERENCES dept(d_code);

ALTER TABLE student_takes_section
ADD FOREIGN KEY (p_id) REFERENCES student(p_id),
ADD FOREIGN KEY (sec_id) REFERENCES section(sec_id);

ALTER TABLE fac_mem_research_areas
ADD FOREIGN KEY (p_id) REFERENCES faculty_member(p_id);
```

```
ALTER TABLE research_assistant
ADD FOREIGN KEY (p_id) REFERENCES faculty_member(p_id);

ALTER TABLE course_keyword
ADD FOREIGN KEY (c_code) REFERENCES course(c_code);

ALTER TABLE section
ADD FOREIGN KEY (c_code) REFERENCES course(c_code),
ADD FOREIGN KEY (instr_p_id) REFERENCES instructor(p_id);

ALTER TABLE research_assistant_assists_section
ADD FOREIGN KEY (p_id) REFERENCES faculty_member(p_id),
ADD FOREIGN KEY (sec_id) REFERENCES section(sec_id);



#1,2,3... kursun toplam kaç keywordu var hesaplar

CREATE VIEW university_db_schema.keyword_count_table
AS
SELECT c_code, count(*) as keyword_count
FROM course_keyword
GROUP BY c_code;

# lists match rates between instructor and course and then order them
CREATE VIEW university_db_schema.instructor_matches_course (Course, Instructor_Id, Match_Rate)
AS
(
   SELECT Course, Instructor_Id, match_count / keyword_count as Match_Rate
   FROM
         (  SELECT c_code as Course, p_id as Instructor_Id, COUNT(*) as match_count
            FROM fac_mem_research_areas JOIN course_keyword ON keyword = research_area
            GROUP BY p_id, c_code
         ) as temp_table
         JOIN
         keyword_count_table
         ON Course = c_code
   ORDER BY c_code ASC, Match_Rate DESC
);
```

8. Populate the database you just created again using SQL script file loaded with sample tuples. The initial database should have the tuples of our department. (The tables should have enough tuples for the SELECT statements to be run accordingly.)

```
SET foreign_key_checks = 0;

INSERT INTO dept (d_code, d_name, d_country, d_office, d_phone, admin_col_name, chair_id, chair_start_date)
VALUES
(1, 'Department of Computer Science', 'Turkiye', 'dept 1 d office', '123-123-123', 'deneme college', 1, '2022-12-26'),
(2, 'Department of Artificial Intelligence', 'Turkiye', 'dept 2 d office', '444-555-666', 'deneme2 college', 2, '2020-11-11'),
(3, 'Department of Software Engineering ', 'Turkiye', 'dept 3 d office', '111-555-666', 'deneme3 college', 3, '1999-11-11');

INSERT INTO faculty_member (p_id, f_name, l_name, address,phone,d_code,e_office,msc,phd)
VALUES
(1, 'professor f_name', 'professor l_name', 'professor address', '555-555-555', 1, 'professor e_office', 'professor msc', 'professor phd'),
```

```
(2, 'associate professor f_name', 'associate professor l_name', 'associate professor address', '444-444-444', 1,
'associate professor e_office', 'associate professor msc', 'associate professor phd'),
(3, 'assistant professor f_name', 'assistant professor l_name', 'assistant professor address', '333-333-333', 1,
'assistant professor e_office', 'assistant professor msc', 'assistant professor phd'),
(4, 'research assistant f_name', 'research assistant l_name', 'research assistant address', '111-111-111', 1, 'research
assistant e_office', 'research assistant msc', 'research assistant phd'),
(10, 'professor2 f_name', 'professor2 l_name', 'professor2 address', '222-555-555', 2, 'professor2 e_office',
'professor msc', 'professor phd'),
(11, 'associate professor2 f_name', 'associate professor2 l_name', 'associate professor2 address', '222-444-444', 2,
'associate professor2 e_office', 'associate professor2 msc', 'associate professor2 phd'),
(12, 'assistant professor2 f_name', 'assistant professor2 l_name', 'assistant professor2 address', '222-333-333', 3,
'assistant professor2 e_office', 'assistant professor2 msc', 'assistant professor2 phd'),
(13, 'research assistant2 f_name', 'research assistant2 l_name', 'research assistant2 address', '222-111-111', 3,
'research assistant2 e_office', 'research assistant2 msc', 'research assistant2 phd');

INSERT INTO instructor (p_id, prof_type)
VALUES
(1, 'PROFESSOR'),
(2, 'ASSOCIATE PROFESSOR'),
(3, 'ASSISTANT PROFESSOR'),
(10, 'PROFESSOR'),
(11, 'ASSOCIATE PROFESSOR'),
(12, 'ASSISTANT PROFESSOR');

INSERT INTO research_assistant(p_id)
VALUES
(4),
(13);

INSERT INTO college (c_name, c_office, c_phone, dean_id)
VALUES
('deneme college', 'deneme college c office', '555-555-555', 1),
('deneme2 college', 'deneme2 college c office', '111-111-111', 5),
('deneme3 college', 'deneme3 college c office', '222-222-222', 1);

INSERT INTO student (p_id, f_name, l_name, address, phone, d_code, major, dob)
VALUES
(5, 'stud1 f_name', 'stud1 l_name', 'stud1 address', '312-312-312', 1, 'Computer Engineering', '2000-01-01'),
(6, 'stud2 f_name', 'stud2 l_name', 'stud2 address', '312-111-312', 1, 'Computer Science', '2000-01-01'),
(7, 'stud3 f_name', 'stud3 l_name', 'stud3 address', '312-312-111', 1, 'Software Engineering', '2000-01-01'),
(8, 'stud4 f_name', 'stud4 l_name', 'stud4 address', '111-312-312', 1, 'Mathematical Computer Science', '2000-02-01'),
(9, 'stud9 f_name', 'stud9 l_name', 'stud9 address', '999-312-312', 2, 'Software Engineering', '2000-02-01'),
(14, 'stud6 f_name', 'stud6 l_name', 'stud6 address', '222-222-312', 1, 'Practical Computer Science', '2000-01-01'),
(15, 'stud7 f_name', 'stud7 l_name', 'stud7 address', '777-312-111', 2, 'Software Engineering', '2000-01-01'),
(16, 'stud8 f_name', 'stud8 l_name', 'stud8 address', '888-312-312', 3, 'Artificial Intelligence', '1999-02-01'),
(17, 'stud9 f_name', 'stud9 l_name', 'stud9 address', '999-222-312', 3, 'Artificial Intelligence', '1998-01-01');

INSERT INTO curriculum (cu_id, followed_by_deptCode)
VALUES
(1, 1),
(2, 2),
(3, 3);

INSERT INTO course (c_code, co_name, credits, course_level, c_desc, in_curriculum)
VALUES
(1, 'Maths', 5, 1, 'maths desc', 1),
(2, 'Science', 6, 2, 'science desc', 1),
(3, 'Music', 2, 1, 'music desc', 1),
(4, 'Object Oriented Programing', 6, 2, 'OOP desc', 2),
(5, 'Artificial Intelligence', 2, 1, 'AI desc', 3);

INSERT INTO mandatory_course (c_code)
VALUES (1),(4),(5);
INSERT INTO optional_course (c_code, is_technical)
VALUES (2, true), (3, false);
```

```
INSERT INTO section (sec_id, sec_no, bldg, room_no, DaysTime, sec_year, sem, c_code, instr_p_id)
VALUES
(1, 123, 'Building A', 38567, 'MW 11:30 am', 2022, 'Fall', 1, 1),
(2, 123, 'Building B', 13867, 'F 11:30 am', 2022, 'Fall', 2, 2),
(3, 123, 'Building C', 33867, 'M 09:30 am', 2022, 'Fall', 3, 3);

INSERT INTO student_takes_section (p_id, sec_id, grade)
VALUES
(5, 1, 78), (5, 2, 78), (5, 3, 78),
(6, 1, 100), (6, 2, 100), (6, 3, 100),
(7, 1, 22), (7, 2, 22), (7, 3, 27),
(8, 1, 71), (8, 2, 71), (8, 3, 71),
(9, 1, 5),
(14, 1, 51), (14, 2, 71), (14, 3, 11),
(15, 1, 61), (15, 2, 21), (15, 3, 22),
(16, 1, 21), (16, 2, 55), (16, 3, 33),
(17, 1, 81), (17, 2, 66), (17, 3, 44);

INSERT INTO research_assistant_assists_section (p_id, sec_id)
VALUES (4, 1),(4, 2),(4, 3);

SET foreign_key_checks = 1;

INSERT INTO course_keyword (c_code, keyword)
VALUES (1, 'kw11'),(1, 'kw12'),(1, 'kw13'),
       (2, 'kw21'),(2, 'kw22'),
       (3, 'kw31'),(3, 'kw32'),(3, 'kw33'),(3, 'kw34'),
       (4, 'kw41'),(1, 'kw42'),(1, 'kw43'),
       (5, 'kw51'),(2, 'kw52');

INSERT INTO fac_mem_research_areas (p_id, research_area)
VALUES (1, 'kw11'),(2, 'kw12'),(3, 'kw13'),
       (1, 'kw21'),(3, 'kw22'),(1, 'kw12'),
       (2, 'kw31'),(1, 'kw32'),(4, 'kw33'),(4, 'kw34'),
       (10, 'kw41'),(11, 'kw42'),(12, 'kw43'),(13, 'kw44')
;

#UPDATE Statements
UPDATE student
SET d_code = '2', major = 'Software Engineering'
WHERE d_code = '3';

UPDATE course_keyword
SET keyword = '53'
WHERE c_code = '5' AND keyword = "kw51";

UPDATE fac_mem_research_areas
SET research_area = 'kw45'
WHERE p_id = '10';

#DELETE Statements
DELETE FROM student_takes_section WHERE p_id = '17' AND sec_id = '2';
DELETE FROM student_takes_section WHERE p_id = '7'AND grade = '27';
DELETE FROM course_keyword WHERE c_code = '5';
DELETE FROM fac_mem_research_areas WHERE p_id = '10';
```

9. Write down 3 triggers for 3 different tables. Triggers should be meaningful.

```
CREATE TABLE STUDENT_LOGS (
   message VARCHAR(100) PRIMARY KEY
);
```

```
DELIMITER $$
CREATE TRIGGER Section_population_del BEFORE DELETE ON student_takes_section
FOR EACH ROW
BEGIN
        IF EXISTS(
                SELECT COUNT(*)
                FROM student_takes_section
                GROUP BY sec_id
                HAVING COUNT(*) <= 5
        )
  THEN
                SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Section must have at least 5 students. Cannot perform this delete
command.';
   END IF;
END$$
DELIMITER ;

DELIMITER //
CREATE TRIGGER log_student_ins AFTER INSERT ON STUDENT
FOR EACH ROW BEGIN
  INSERT INTO student_logs (message) VALUES (" New student added.");
END;
//
DELIMITER ;

INSERT INTO student (p_id, f_name, l_name, address, phone, d_code, major, dob)
VALUES
(5123123, 'stud1 f_name', 'stud1 l_name', 'stud1 address', '312-312-312', 1, 'Computer Engineering', '2000-01-01');

DELIMITER //
CREATE TRIGGER log_dept_ins AFTER INSERT ON dept
FOR EACH ROW BEGIN
  INSERT INTO student_logs (message) VALUES ("New dept was added. Inform the department chair.");
END;
//
DELIMITER ;
```

10. Write down 3 check constraints and 3 assertions. Check constraints and assertions should be meaningful.
Check constraints:

```
ALTER TABLE student_takes_section
ADD CHECK (grade>=0);

ALTER TABLE course
ADD CHECK (credits>=0);

ALTER TABLE section
ADD CHECK (room_no>=0);
```

Assertions:

```
#research_assistant - instructor
DELIMITER $$
CREATE TRIGGER research_assistant_trig_ins BEFORE INSERT
ON research_assistant
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM instructor
        WHERE NEW.p_id = instructor.p_id
    )
    THEN
```

```
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING - RESEARCH ASSISTANT EXISTS IN
INSTRUCTOR TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER research_assistant_trig_upd BEFORE UPDATE
ON research_assistant
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM instructor
        WHERE NEW.p_id = instructor.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'UPDATE WARNING - RESEARCH ASSISTANT EXISTS IN
INSTRUCTOR TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER instructor_ins BEFORE INSERT
ON instructor
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM research_assistant
        WHERE NEW.p_id = research_assistant.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING - INSTRUCTOR EXISTS IN
RESEARCH_ASSISTANT TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER instructor_upd BEFORE UPDATE
ON instructor
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM research_assistant
        WHERE NEW.p_id = research_assistant.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'UPDATE WARNING - INSTRUCTOR EXISTS IN
RESEARCH_ASSISTANT TABLE';
    END IF;
END$$
DELIMITER ;
```

```
# mandatory - optional
DELIMITER $$
CREATE TRIGGER mandatory_ins BEFORE INSERT
ON mandatory_course
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM optional_course
        WHERE NEW.c_code = optional_course.c_code
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING - COURSE EXISTS IN OPTIONAL COURSE
TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER mandatory_upd BEFORE UPDATE
ON mandatory_course
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM optional_course
        WHERE NEW.c_code = optional_course.c_code
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'UPDATE WARNING - COURSE EXISTS IN OPTIONAL COURSE
TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER optional_ins BEFORE INSERT
ON optional_course
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM mandatory_course
        WHERE NEW.c_code = mandatory_course.c_code
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING - COURSE EXISTS IN MANDATORY COURSE
TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER optional_ins BEFORE UPDATE
ON optional_course
```

```
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM mandatory_course
        WHERE NEW.c_code = mandatory_course.c_code
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'UPDATE WARNING - COURSE EXISTS IN MANDATORY COURSE
TABLE';
    END IF;
END$$
DELIMITER ;

# faculty_member - student
DELIMITER $$
CREATE TRIGGER faculty_member_ins BEFORE INSERT
ON faculty_member
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM student
        WHERE NEW.p_id = student.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING - FACULTY MEMBER EXISTS IN STUDENT
TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER faculty_member_upd BEFORE UPDATE
ON faculty_member
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM student
        WHERE NEW.p_id = student.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'UPDATE WARNING - FACULTY MEMBER EXISTS IN STUDENT
TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER student_ins BEFORE INSERT
ON student
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
```

```
        FROM faculty_member
        WHERE NEW.p_id = faculty_member.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'INSERT WARNING - STUDENT EXISTS IN FACULTY MEMBER
TABLE';
    END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER student_upd BEFORE UPDATE
ON student
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT *
        FROM faculty_member
        WHERE NEW.p_id = faculty_member.p_id
    )
    THEN
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'UPDATE WARNING - STUDENT EXISTS IN FACULTY MEMBER
TABLE';
    END IF;
END$$
DELIMITER ;
```

11. Write down the following SQL statements:
    a. Write sample INSERT, DELETE and UPDATE statements for 3 of the tables you have chosen.

```
INSERT INTO faculty_member (p_id, f_name, l_name, address,phone,d_code,e_office,msc,phd)
VALUES
(1, 'professor f_name', 'professor l_name', 'professor address', '555-555-555', 1, 'professor e_office', 'professor
msc', 'professor phd');

INSERT INTO instructor (p_id, prof_type)
VALUES
(12, 'ASSISTANT PROFESSOR');

INSERT INTO research_assistant(p_id)
VALUES
(13);
```

```
#UPDATE Statements
UPDATE student
SET d_code = '2', major = 'Software Engineering'
WHERE d_code = '3';

UPDATE course_keyword
SET keyword = '53'
WHERE c_code = '5' AND keyword = "kw51";

UPDATE fac_mem_research_areas
SET research_area = 'kw45'
WHERE p_id = '10';
```

```
#DELETE Statements
DELETE FROM student_takes_section WHERE p_id = '17' AND sec_id = '2';
DELETE FROM student_takes_section WHERE p_id = '7'AND grade = '27';
DELETE FROM course_keyword WHERE c_code = '5';
DELETE FROM fac_mem_research_areas WHERE p_id = '10';
```

b. Write 10 SELECT statements for the database you have implemented.

i. 3 of them should use just one table.

```
SELECT dean_id
FROM college
WHERE c_name = "deneme college";

SELECT *
FROM course
WHERE credits > 3;

SELECT *
FROM curriculum;
```

ii. 4 of them should use a minimum of 2 tables.

```
SELECT *
FROM dept d, curriculum cu
WHERE cu.followed_by_deptCode = d.d_code;

SELECT *
FROM course NATURAL JOIN course_keyword;

SELECT *
FROM college JOIN dept ON college.c_name = dept.admin_col_name;

SELECT *
FROM research_assistant NATURAL JOIN research_assistant_assists_section;
```

iii. 3 of them should use a minimum of 3 tables.

```
SELECT *
FROM college co, dept d, curriculum cu
WHERE co.c_name = d.admin_col_name AND cu.followed_by_deptCode = d.d_code;

SELECT c.c_code, count(*)
FROM course c NATURAL JOIN section NATURAL JOIN student_takes_section
GROUP BY c.c_code;

SELECT *
FROM faculty_member f, dept d, college c
WHERE f.d_code = d.d_code AND d.admin_col_name=c.c_name;
```

c. Write 5 original SELECT statements that you think critical to interaction and integration points for the database.

```
#avg number of research areas per instructor type
SELECT instr.prof_type, count(*) / count(distinct instr.p_id) AS "Average Number of Research Areas"
FROM fac_mem_research_areas ra NATURAL JOIN instructor instr NATURAL JOIN faculty_member fm
GROUP BY instr.prof_type
ORDER BY count(distinct instr.p_id)/count(*);

# Müfredat numarası 1'e kayıtlı olan ve zorunlu alınması gereken derslerden o derse özel toplam keyword sayısına ve o dersin açıklamasına ulaşılmak istenmiştir.
SELECT C.c_code, K.in_curriculum,  count(*) as keyword_count, c_desc
FROM university_db_schema.course_keyword AS C,university_db_schema.mandatory_course AS M ,
university_db_schema.course AS K
WHERE C.c_code = M.c_code AND C.c_code = K.c_code AND K.in_curriculum = "1"
GROUP BY C.c_code;


# 2022 yılında 1 numaralı sectionı alan öğrencilerden ders notlarına göre ortalamada ilk 3'e girenlere ödül verilmek üzere adresleri öğrenilmek istenmiştir.
SELECT S.p_id,S.sec_id, AVG(S.p_id) AS Average_Grade, U.address
FROM university_db_schema.student_takes_section AS S,university_db_schema.section AS K,
university_db_schema.student AS U
WHERE S.sec_id = "1" AND K.sec_year = "2022" AND S.p_id = U.p_id
GROUP BY S.p_id,S.sec_id
ORDER BY AVG(S.p_id) DESC
LIMIT 3;

# Yüksek lisans ve doktora tezi olan faculty memberların yanına 1, diğerlerinin yanına 0 yazdır
SELECT FM.f_name,
  CASE
    WHEN FM.phd IS NOT NULL AND FM.phd IS NOT NULL THEN 1
    ELSE 0
  END as has_theses
FROM university_db_schema.faculty_member AS FM;

# Aynı dekanın verdiği birden fazla kurs varsa bu kursları yazdır
SELECT DISTINCT c1.c_name AS SAME_DEAN_COURSES, FM.f_name AS DEAN_NAME
FROM university_db_schema.college c1
JOIN university_db_schema.college c2 ON c1.dean_id = c2.dean_id ,
university_db_schema.faculty_member AS FM
WHERE c1.c_name <> c2.c_name AND c1.dean_id = FM.p_id ;


# Kredisi 5 veya daha fazla olan derslere 1, diğerlerinin yanına 0 yazdır
SELECT C.co_name,
  CASE
    WHEN C.credits >= 5 THEN 1
    ELSE 0
  END as has_credits_greater_than_4
FROM university_db_schema.course AS C;
```